

CMPT 321

Introduction to databases

Fall 2017

Marina Barsky

The data world

- We aggressively acquire data and keep it forever
- The real freedom is when we have access to all the data
- The goal – preserve knowledge and make it accessible to everyone
- More ambitious goal – enough data that we can simulate real world and understand its behaviour from these simulations

Types of data: music

The screenshot displays the iTunes Store interface with a sidebar on the left and a main content area. The sidebar includes sections for LIBRARY (Music, Movies, TV Shows, Podcasts, Radio), STORE (iTunes Store, Purchased), and PLAYLISTS (Party Shuffle, classical, Jazz, Kid's Music, Not Kid's Music, Recently Played, sesame street, songs, Top 25 Most Pla..., adene, Ari, Baby Ella: One, Bill Cosby: Himself, Brookline Fun, Ecuador Shhh, Give Him the Oo..., gloom, gray, Jason, Joe Hellerstein's ..., lud's iTunes).

The main content area features a top navigation bar with "20 Songs" and the email "hellerstein@cs.berkeley.edu". Below this are three promotional banners: "ROBOT CHICKEN. New Season", "John Lennon Catalog + Exclusive Videos Just Added", and "Amy Grant My Favorites, Exclusive Collection & Catalog Just Added".

The central section is titled "iTunes STORE" and "NEW RELEASES". It has tabs for Music, Movies, TV Shows, Jazz, Rock, and Classical. The "NEW RELEASES" section displays a grid of new releases:

- High School Musical 2 - ... Various Artists
- Working Class Hero - T... John Lennon
- The Storyteller Collection Amy Grant
- Singularity Various Artists
- Halfway to Hazard Halfway to Hazard
- Unglamorous (Bonus Tr... Lori McKenna
- Some Mad Hope Matt Nathanson
- Live At Radio City Dave Matthews Band & T...

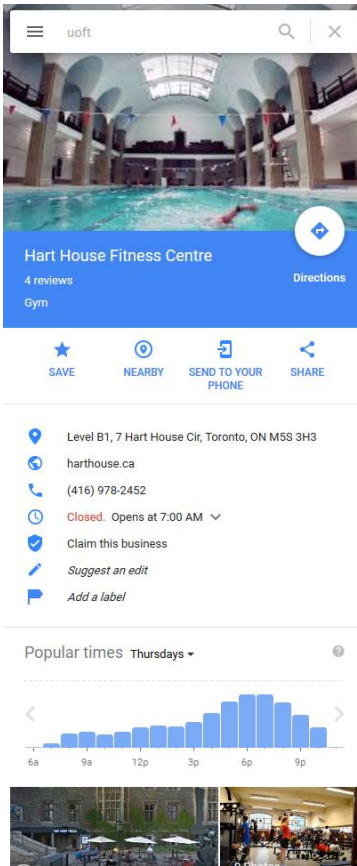
Below the new releases, there are several promotional tiles:

- "High School Musical Kids & Family"
- "A Fish Called Wanda Comedy"
- "Zoolander Comedy"
- "Wild Hogs Comedy"
- "Renaissance Sci-Fi & Fantasy"
- "IGGY OPEN UP AND BLEED New in Audiobooks"
- "A Fine Frenzy SINGLE OF THE WEEK FREE"
- "iTunes Guide to HIGH SCHOOL MUSICAL 2"
- "The perfect fit for your iPod. Shop by iPod type"
- "FREE TV"
- "best of the store AUG 14, 2007"


On the right side, there are "QUICK LINKS" and "TOP SONGS" sections. The "QUICK LINKS" section includes: Browse, Power Search, Account, Buy iTunes Gifts, Redeem, Support, My Alerts (NEW), Complete My Album (NEW), and iTunes Plus (NEW). The "TOP SONGS" section lists:

- Beautiful Girls Sean Kingston
- Stronger Kanye West
- Me Love Sean Kingston
- The Way I Are Timbaland featuring Keri Hi...
- Ayo Technology 50 Cent

Types of data: geodata



uoft




Hart House Fitness Centre
4 reviews
Gym

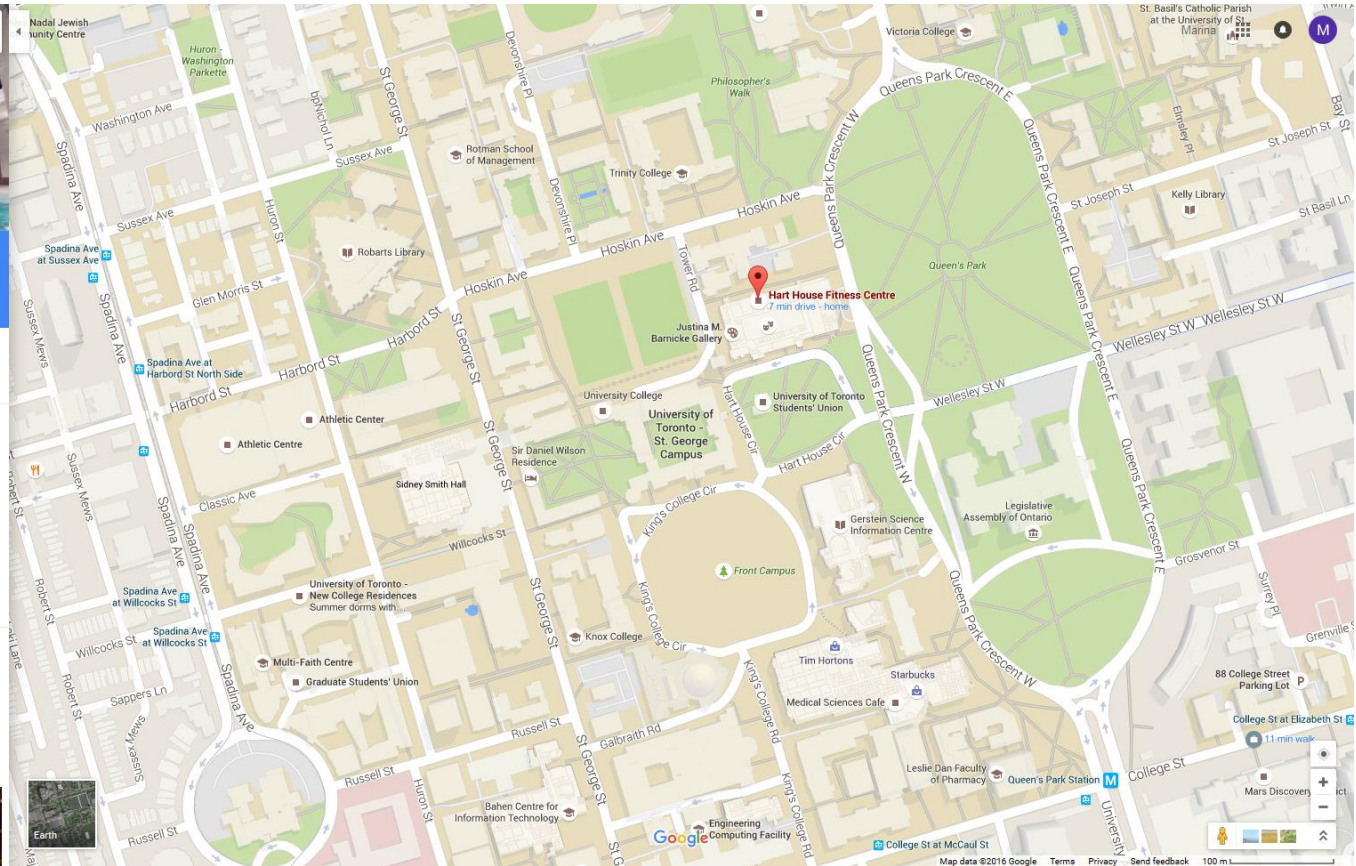

SAVE NEARBY SEND TO YOUR PHONE SHARE

Level B1, 7 Hart House Cir, Toronto, ON M5S 3H3
harthouse.ca
(416) 978-2452
Closed. Opens at 7:00 AM
Claim this business
Suggest an edit
Add a label

Popular times Thursdays



| Time | Popularity |
|------|------------|
| 6a | Low |
| 9a | Low |
| 12p | Low |
| 3p | Low |
| 6p | High |
| 9p | Low |



Types of data: bio-sequences

The screenshot displays the NCBI Map Viewer interface. At the top, the NCBI logo is on the left, and the 'NCBI Map Viewer' title with a compass icon is in the center. Below this is a navigation bar with tabs for 'PubMed', 'Nucleotide', 'Protein', 'Genome', 'Gene', 'Structure', 'PopSet', 'Taxonomy', and 'Help'. A search bar is present with the text 'Search for' and a dropdown menu set to 'on chromosome(s)'. A 'Find' button and an 'Advanced Search' link are also visible.

The main content area is titled '*Homo sapiens (human) genome view*' and includes links for 'Build 36.2 statistics' and 'Switch to previous build'. A 'BLAST search the human genome' link is located on the right.

The central part of the page shows a karyotype of the human genome, with chromosomes arranged in two rows. The first row contains chromosomes 1 through 13, and the second row contains chromosomes 14 through 22, X, Y, and MT. Each chromosome is represented by a vertical bar with a centromere.

On the left side, there is a blue sidebar with a 'Map Viewer' section containing links for 'Map Viewer Home', 'Map Viewer Help', 'Human Maps Help', and 'Release Notes'. Below this is the 'NCBI Resources' section with links for 'Genome Project', 'TaxPlot', 'Consensus CoDing Sequence (CCDS)', 'Human Genome Resources', 'NCBI Handbook', and 'RefSeq'. At the bottom of the sidebar is the 'Organism Data in GenBank' section with a link for 'Whole Genome Association (WGA)'.

At the bottom of the main content area, there is a blue box containing the following text: '**Lineage:** [Eukaryota](#); [Metazoa](#); [Chordata](#); [Craniata](#); [Vertebrata](#); [Euteleostomi](#); [Mammalia](#); [Eutheria](#); [Euarchontoglires](#); [Primates](#); [Haplorrhini](#); [Catarrhini](#); [Hominidae](#); [Homo](#); [Homo sapiens](#)'.

Below the lineage box, there is a paragraph of text: '**September 2006:** NCBI released an annotation update for the human genome (NCBI Build 36.2); this update does not change the genome assembly. The previous version of the genome assembly, [NCBI Build 35.1](#), can still be accessed for Map Viewer display and for BLAST. For additional information about changes, statistics, and the status of the CCDS project please refer to:

“Big data”?

- Basic demographic information—age, sex, income, ethnicity, language, religion, housing status, and location—of every living human being on the planet can be stored in 100GB
- This would create a table of 6.75 billion rows and 10 columns.
- Should that be considered “big data”?

From “Pathologies of Big Data” Article by Adam Jacobs in the ACM Communications, August 2009.

Data Units

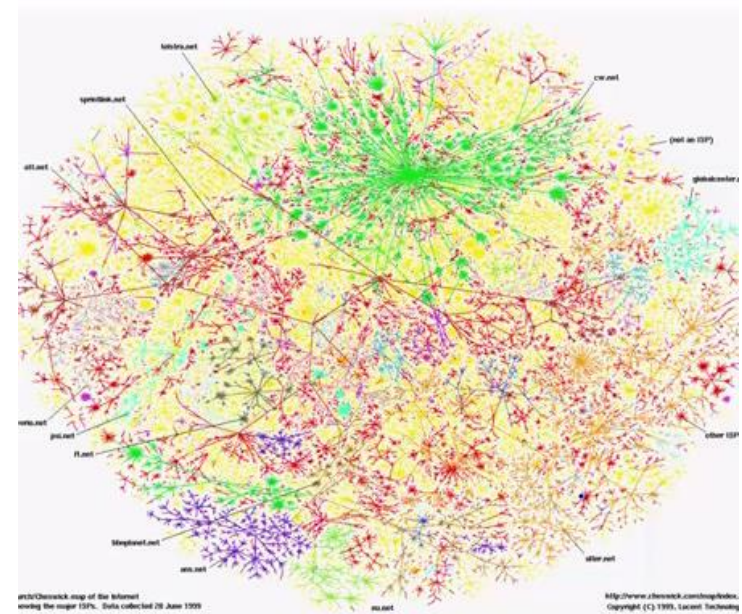
Roughly:

| | | | |
|---|------|----------|-----------|
| K | Kilo | 2^{10} | 10^3 |
| M | Mega | 2^{20} | 10^6 |
| G | Giga | 2^{30} | 10^9 |
| T | Tera | 2^{40} | 10^{12} |
| P | Peta | 2^{50} | 10^{15} |

Example: Volume

- The web
 - 20+ billion web pages x 20KB = **400+ TB**
 - One computer can read 30-35 MB/sec from one disk – 4 months just to read the web

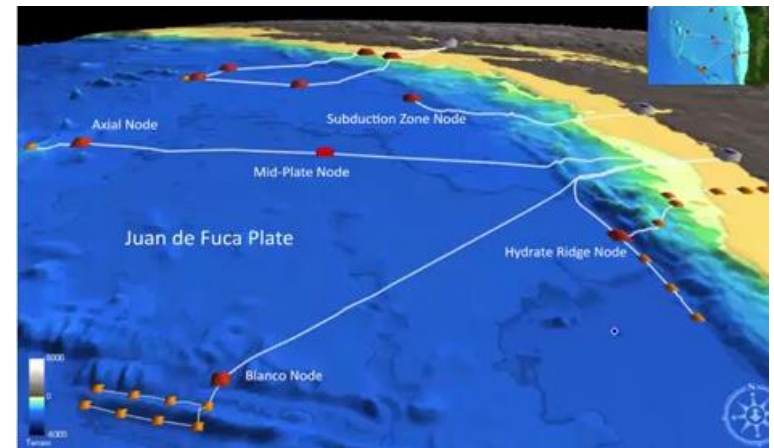
The web



Example: Variety

- NSF Ocean Observatories Initiative
 - Data is collected from satellites, vessels, sensors
 - 1000 km of optic cable on the seafloor with thousands of chemical, physical, biological sensors
 - 50 TB/year of different data types

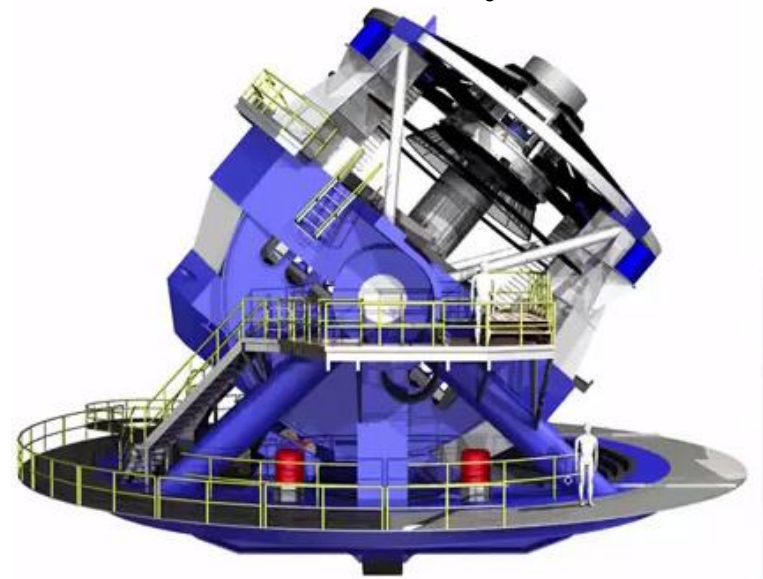
Ocean Sciences



Example: Velocity

- Large Synoptic Survey Telescope (LSST)
 - 40 TB/day
 - 40+ PB in its 10 year lifetime
 - 400 mbps sustained data exchange rate between Chile and NSCA
- Largest database in the world: World Data Centre for Climate (WDCC):
 - 100 TB of sensor data/year
 - 110 TB of simulation data/year
 - 6PB of additional information stored on tapes

Astronomy



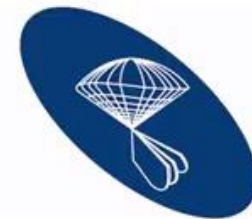
Big Data: 4V

- **V**olume
- **V**ariety
- **V**elocity

- **V**eracity: can we trust this data?

Evolution of Science

- **Empirical Science** – collect and systematize facts
- **Theoretical Science** – formulate theories and empirically test them
- **Computational Science** – run automatic proofs, simulations
- **e-Science (Data Science)** – collect data without clear goal - and test theories, find patterns **in the data itself**



SLOAN DIGITAL SKY SURVEY

Science is about asking questions

Traditionally: “Query the world”

Data acquisition for a specific hypotheses

Data science: “Download the world”

Data acquired en masse in support of future hypotheses

Computational challenge

The cost of data **acquisition** has dropped

The cost of **processing, integrating** and **analyzing** data is the new bottleneck

“...the necessity of grappling with Big Data, and the desirability of unlocking the information hidden within it, is now a key theme in all the sciences – arguably the key scientific theme of our times”

F. Diebold

Efficient data manipulation

Poll: How much time modern scientists spend “**handling data**” as opposed to “**doing science**”?

Mode answer: **90%**

“the Next Wave of InfraSress” (J. Mashey)

Raw data \neq knowledge!

We need to store data in a system that provides:

- **Non-volatile reliable** storage
- Organized for **efficient** queries of any kind

Is a **File System** a candidate?

Thought Experiment 1:

- You and your project partner are editing the same file.
- You both save it at the same time.
- Whose changes survive?

A) Yours **B) Partner's** **C) Both** **D) Neither** **E) ???**

Is a **File System** a candidate?

Thought Experiment 2:

- You're updating a file.
- The power goes out.
- Which changes survive?

A) All **B) None** **C) All Since Last Save** **D) ???**

Is a **File System** a candidate?

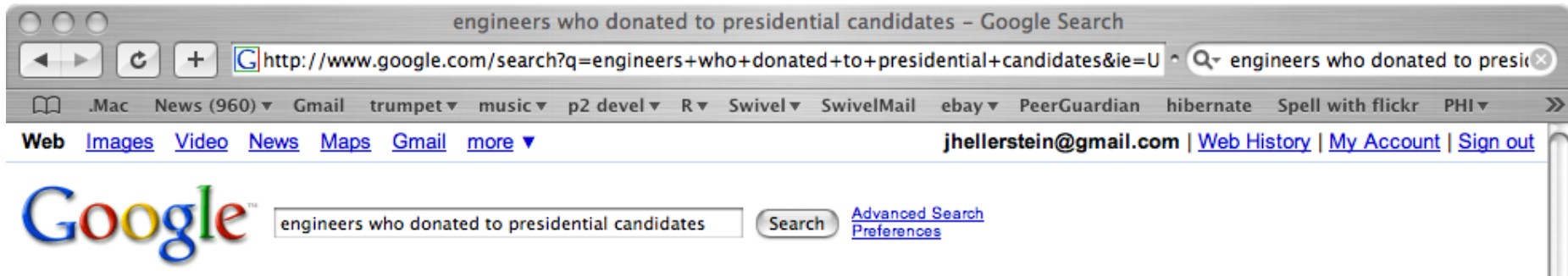
Q: How do you write programs over a subsystem when it promises you only “???” ?

Is the WWW a candidate?

- Crawler indexes pages on the web and we can search for pages by keyword
- Source data is mostly “prose”: *unstructured* and untyped
- Public interface is *search only*:
 - can’t modify the data
 - can’t get summaries, complex combinations of data
- Few guarantees provided for *freshness of data*, *consistency* across data items, *fault tolerance*, ...

“Search” vs. Query

- Try *actors who donated to presidential candidates* in your favorite search engine.
- Now try *engineers who donated to presidential candidates*

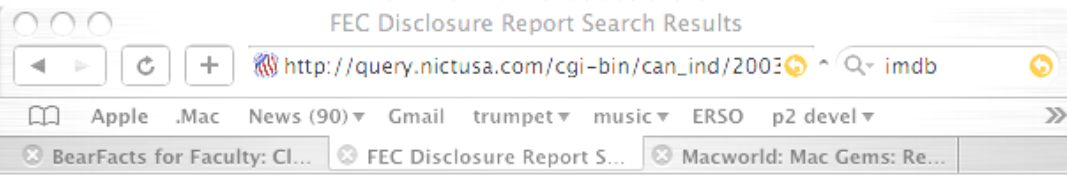
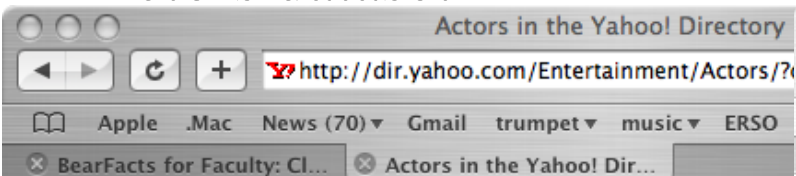


If it isn't “structured”, it can't be searched!

A “Database Query” Approach

Actors dataset

Donors dataset



- [Acting@](#)
- [Books@](#)
- [Fanlistings@](#)
- [Organizati](#)
- [Resumes@](#)
- [Web Direc](#)

SITE LISTINGS [By Popularity](#) | [Alphabetical](#) ([What's This?](#))

[[A](#)][[B](#)][[C](#)][[D](#)][[E](#)][[F](#)][[G](#)][[H](#)][[I](#)][[J](#)][[K](#)][[L](#)][[M](#)][[N](#)][[O](#)][[P](#)][[Q](#)][[R](#)][[S](#)][[T](#)][[U](#)][[V](#)][[W](#)][[X](#)][[Y](#)][[Z](#)]
[[Last \(Z\)](#)] [[Next \(B\)](#)]

- [Aames, Willie \(2\)](#)
[dir.yahoo.com/.../Actors/Aames__Willie](#)
- [Abbott, Bud \(1895-1974\) and Lou Costello \(1906-1](#)
[dir.yahoo.com/.../Comedy_Groups/Abbott__Bud__1895_1974__e](#)
- [Abdul-Jabbar, Kareem@](#)
[dir.yahoo.com/.../Former_Players/Abdul_Jabbar__Kareem](#)
- [Abrahams, Jon \(3\)](#)
[dir.yahoo.com/.../Actors/Abrahams__Jon](#)
- [Acker, Amy \(5\)](#)
[dir.yahoo.com/.../Actors/Acker__Amy](#)
- [Ackles, Jensen \(12\)](#)
[dir.yahoo.com/.../Actors/Ackles__Jensen](#)
- [Acosta, Vanessa](#)
Facts and filmography for the actress whose work includes El Ar
[www.imdb.com/Name?Acosta,+Vanessa](#)

Presented by the Federal Election Commission

Individuals Who Gave To: [KERRY, JOHN F](#)

Sorted By Transaction Type Then Last Name

Committee(s) Used In This Query:

[KERRY-EDWARDS 2004 INC.](#)

[KERRY-EDWARDS 2004 INC. GENERAL ELECTION LEGAL AND ACCOUNTING COMPLIANCE FUND](#)







[JOHN KERRY FOR PRESIDENT, INC](#)


[VETERANS FOR JUSTICE](#)

The query you have chosen matched **222599** individual contributions.


“Yahoo Actors” JOIN “FECInfo”

Federated Facts and Figures - Microsoft Internet Explorer

Address  f.cs.berkeley.edu/demo5.html  Go  Back  Forward  File 


Query Finished 

Results



Q: Did it Work?

| Name | Occupation | Address | Amount |
|---------------------|--------------------------------|-------------------|----------|
| Smits, Jimmy | Self employed | Los Angeles, ... | 250.00 |
| Somers, Suzanne | Self | Valencia, CA ... | 1,000.00 |
| Stamp, Terence | Info Requested | Sanbornville, ... | 1,000.00 |
| Stone, Sharon | Self employed/Actress | Los Angeles, ... | 1,000.00 |
| Streisand, Barbra | Self employed/Singer / Prod... | Santa Monica... | 1,000.00 |
| * Taylor, Elizabeth | Not employed/Homemaker | Tampa, FL 33... | 250.00 |
| Thomas, Heather | CIGNA Healthcare/New Busi... | Nashville, TN ... | 250.00 |
| Thomas, Michelle | | Washington, ... | 300.00 |
| Thomas, Olive | National Council of Church... | Maryville, TN ... | 1,000.00 |
| Thomas, Olive | National Council of Church... | Maryville, TN ... | 1,000.00 |
| Tomlin, Lily | Self employed/Actress | Los Angeles, ... | 250.00 |
| Tripplehorn, Jeanne | Self employed/Actress | Los Angeles, ... | 1,000.00 |
| Wagner, Robert | Self employed/Doctor | McLean, VA 2... | 500.00 |

134 - 

(From Telegraph research group @Berkeley)

To have a real data management system we need to solve problems of:

- **Scale**: data exceeds main memory, specialized (quite complex) EM algorithms, efficiently implemented
- **Sharing**: using the same data by multiple user programs simultaneously (concurrently)
- **Fault-tolerance**: avoiding data loss
- **Consistency**: clean consistent snapshots of data, reinforcing data constraints

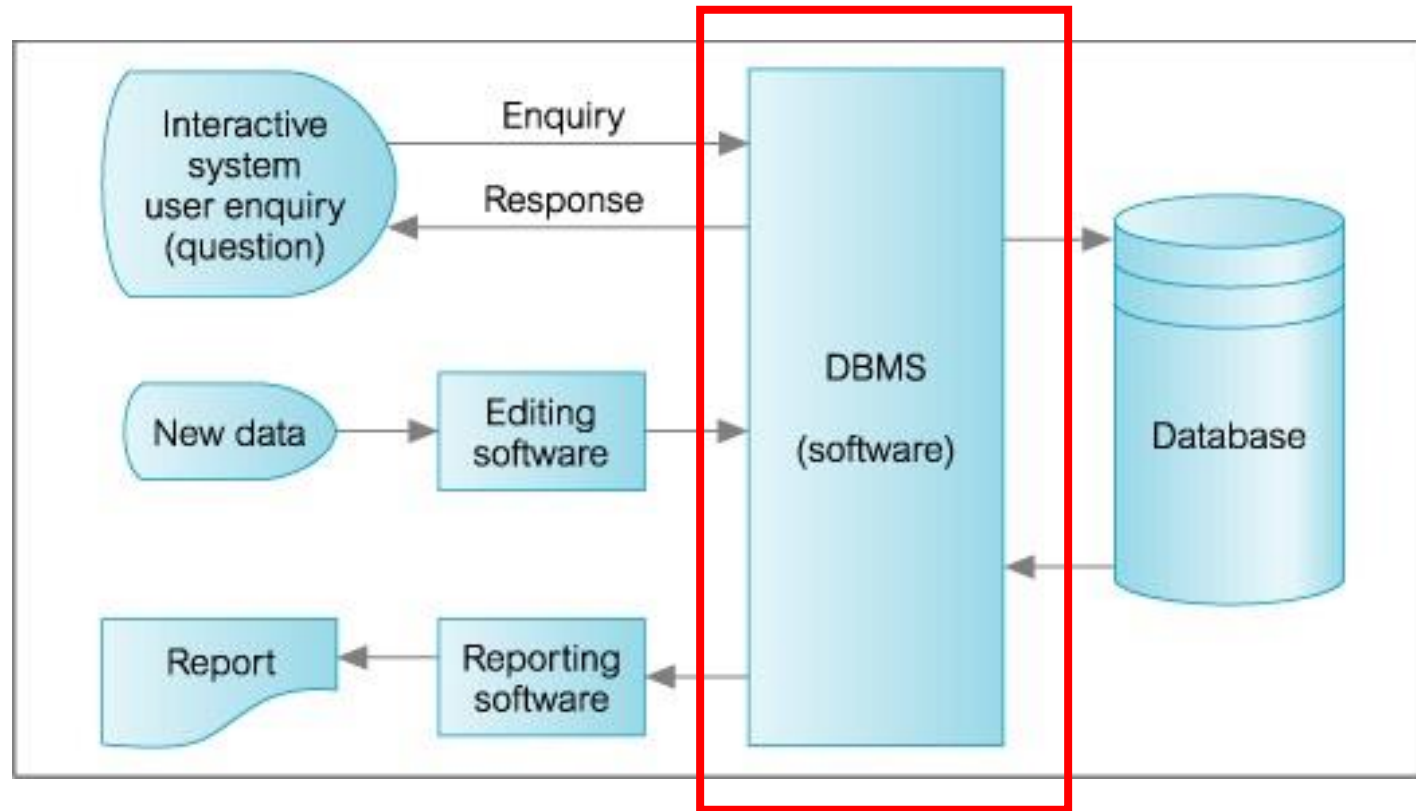
Our dream system:

1. Allows to **create new data collections** and specify their schema (logical structure of the data) in a simple language
2. Enables **data query and modification**, using a simple language
3. Supports **intelligent storage** of very large amounts of data.
 - a. Enforcing constraints (to not allow the insertion of two different people with the same SIN).
 - b. Efficient access to the data for queries and modifications (Indexes).
4. Controls access to data from many users at once (**concurrency**), without allowing “bad” interactions that can corrupt the consistency.
5. **Recovers from** software **failures** and hardware **crashes**.

Such system exists:

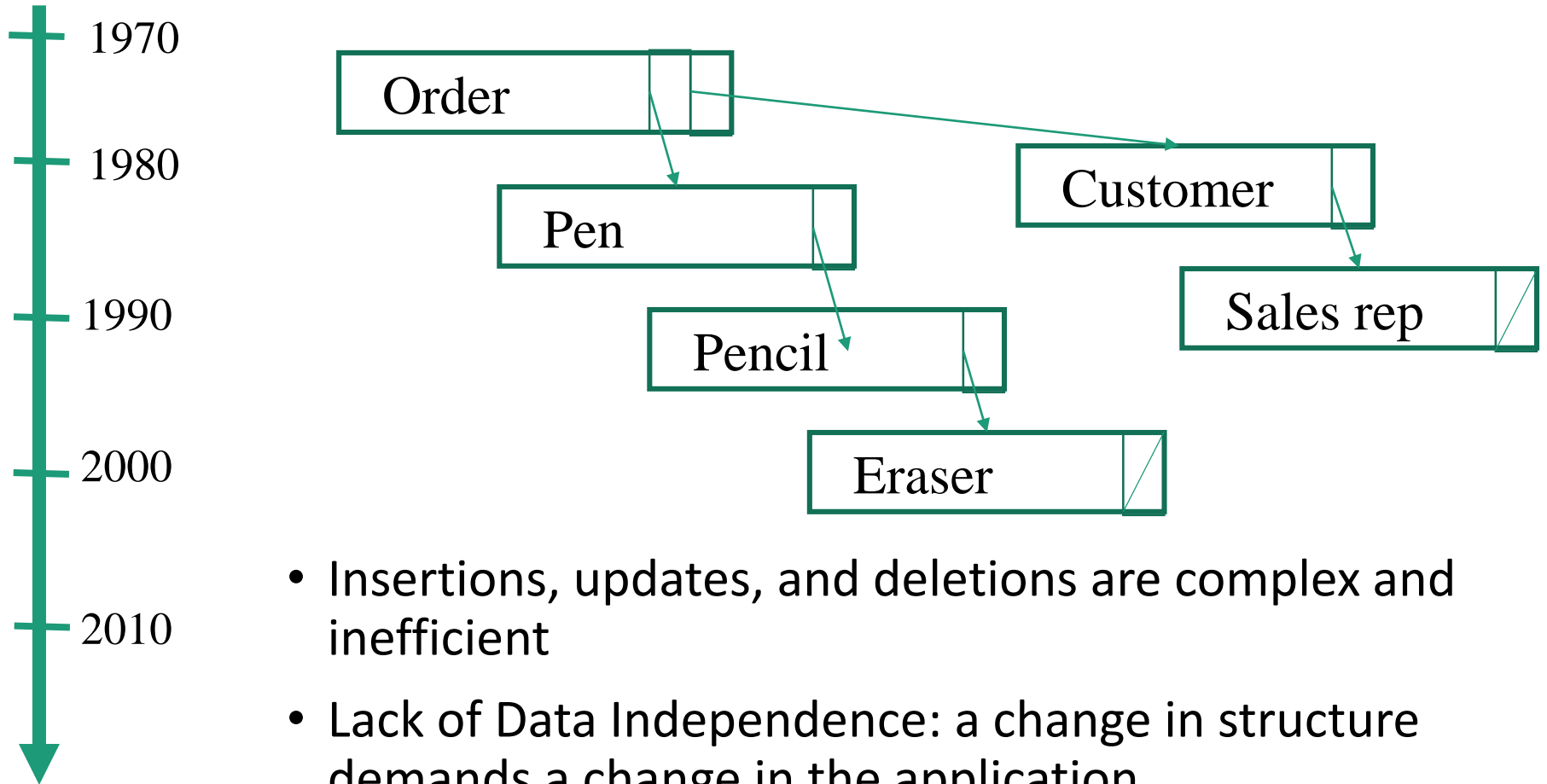
Database Management System (DBMS) - complex *software* for storing and managing databases.

Database management system



History of DBMS

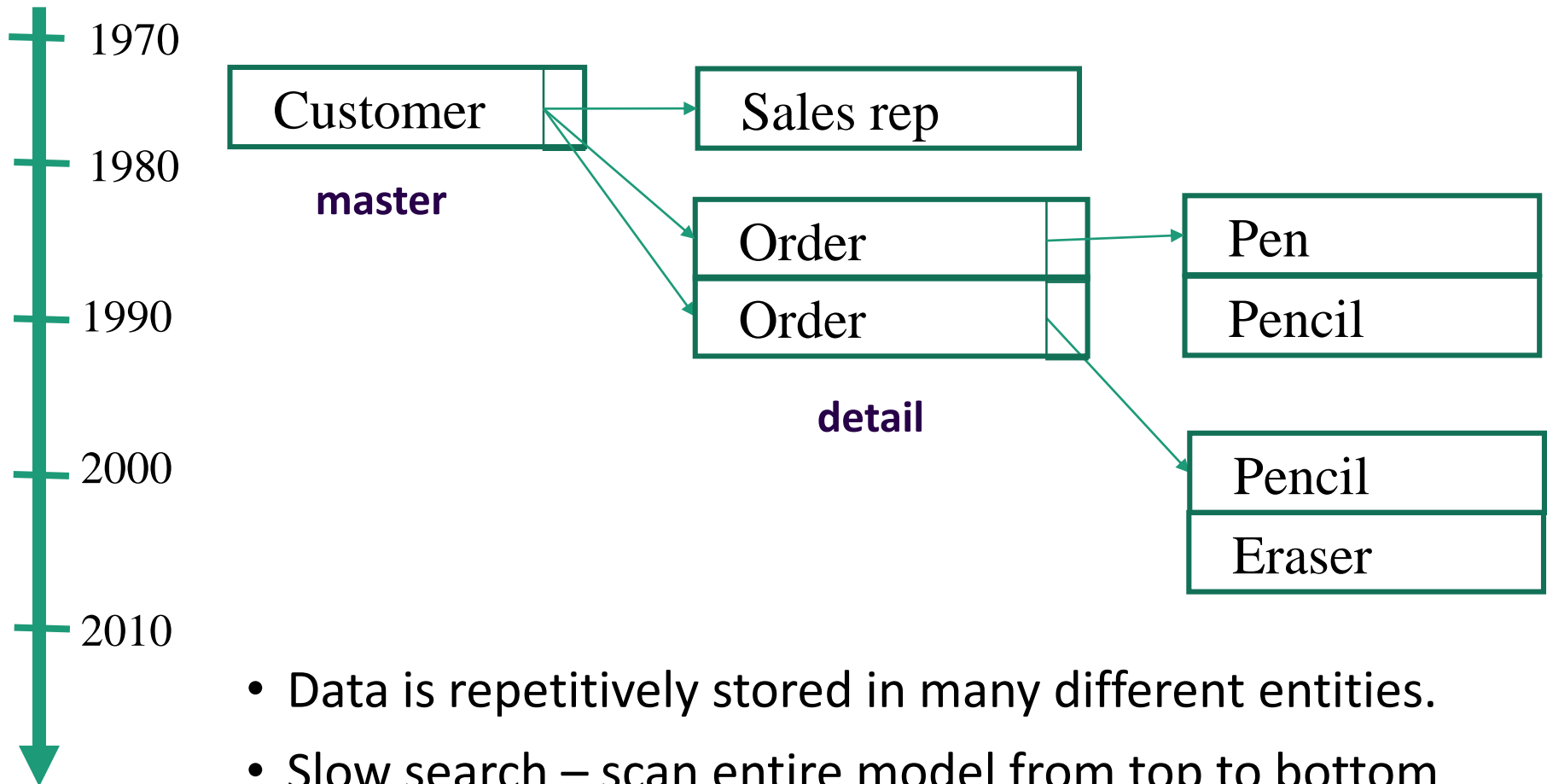
Network databases



- Insertions, updates, and deletions are complex and inefficient
- Lack of Data Independence: a change in structure demands a change in the application
- Unanticipated queries cannot be performed efficiently

History of DBMS

Hierarchical databases



- Data is repetitively stored in many different entities.
- Slow search – scan entire model from top to bottom
- One-to-many relationships only

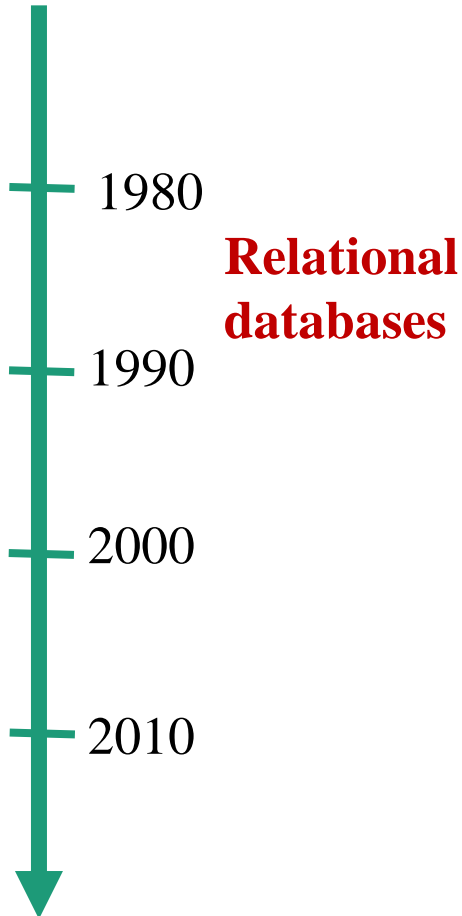
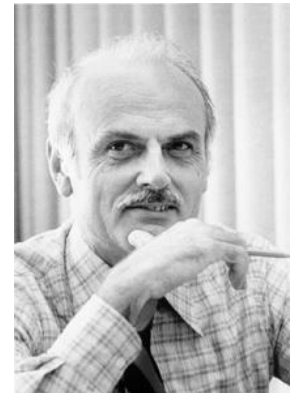
History of DBMS

*God made the integers;
all else is the work of man.*

L. Kronecker, 19-th century
mathematician

*Codd made relations;
all else is the work of man.*

R. Ramakrishnan

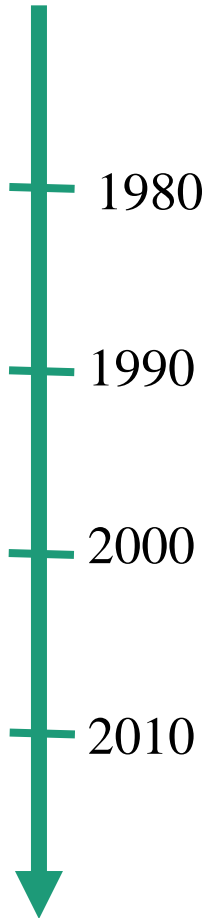


History of DBMS

Think in terms of tables, not bits on disk.

“Activities of users at terminals *should remain unaffected when the internal representation of data is changed.*”

- Pre-relational: if your data changed, your application broke
- Early RDBMSs were buggy and slow, but required only 5% of the application code



**Relational
databases**

Ted Codd's vision

- A database system should present the user with a view of data organized as **tables** (also called *relations*).
- **Behind the scene** there could be a **complex data structure** that allows **rapid** response to a variety of queries. But the user would not be concerned with the storage structure.
- **Queries** could be expressed in a very **high-level language**, which greatly increases the efficiency of database programmers.

Relational databases: key idea

Programs that manipulate tabular data exhibit an *algebraic structure* allowing reasoning and manipulation independently of physical data representation

Can apply relational algebra!



Algebraic optimization: symbolic reasoning on integers

$$N = ((z*2) + ((z*3) + 0))/1$$

Algebraic laws:

1. Identity: $x+0 = x$
2. Identity: $x/1 = x$
3. Distributive: $ax + ay = a*(x+y)$
4. Commutative: $x*y = y*x$

Apply rules 1,3,4,2:

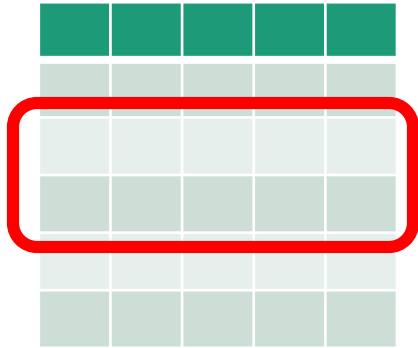
$$N = (2+3)*z$$

One operation instead of five, no division.

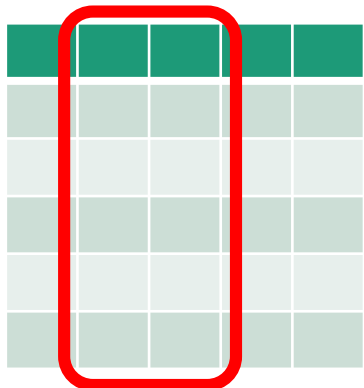
Closure: each operation returns the value of the same type, so operations can be chained

Same idea works with relational algebra!

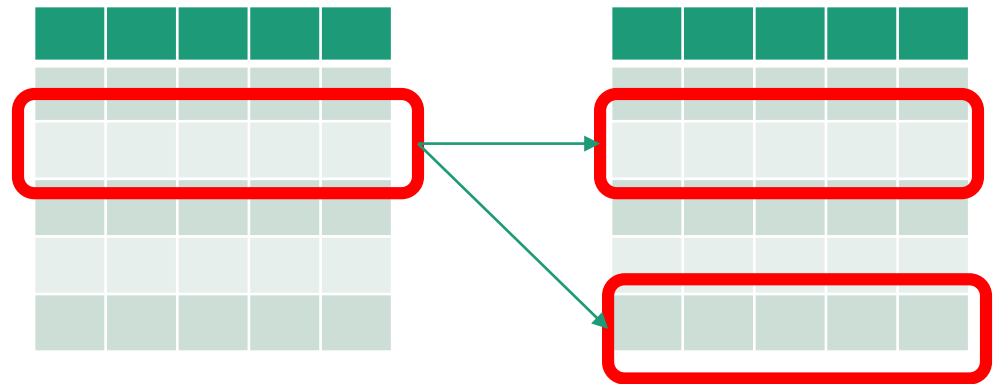
Algebra of tables



Selection σ



Projection π



Join \bowtie

Cross-product \times

Union \cup

Difference $-$

Intersection \cap

Case in favor of **Relational** Database Management Systems

RDBMS provides:

- Physical and logical data independence
- Automatic indexing
- Efficient implementation of RA operators
- Query optimization
- Support and guarantees of atomic transactions

Imagine adding all these features yourself for your next data product!

Early applications of RDBMS's

- Airline reservation systems
- Banking systems
- Corporate records

Data composed of many small **items**, and various **queries** and **modifications** on them.

Example: RDBMS vs Files

- Suppose we have stored in a file called *Employees* records having the fields
(emp_code, name, dept_code)
- and in another file called *Departments* records having the fields:
(dept_code, dept_name)

Suppose now that given an employee, for instance with name "Smith", we want to find out what department is he working for.

Files: solution

In the **absence** of DBMS we have to *write a program* which will:

1. open the file `Employees`
2. declare a variable of the same type as the records stored in the file
3. scan the file:
 - while** the end of the file is not yet encountered,
assign the current record to above variable.
if the value of the name field is `"Smith"` then remember
the value of the `dept_code` field. Suppose it is `"100"`
4. search in a similar way for a record with `"100"` for the `dept_code` in the `Department` file.
5. print the `dept_name` when successfully finding the `dept_code`.

Very painful procedure

Modern RDBMS solution

Compare it to the short and elegant SQL query

```
SELECT      dept_name
FROM        Employees, Department
WHERE       Employees.name="Smith" AND
            Employees.dept_code = Department.dept_code
```


Example: Query optimization

SELECT Accounts.balance

FROM Customers, Accounts

WHERE Customers.SIN = Accounts.SIN **AND** Customers.name = 'Sally';

This query - if executed naively:

- Pairs tuples of tables specified in the **FROM**-clause into a new table **R**.
- Chooses from **R** the tuples satisfying the condition in the **WHERE** clause.
- Produces as answer only the values of attributes in **SELECT**-clause.

The performance would be terrible, because of the usually enormous (quadratic) size of all pairs of tuples.

Example: Query optimization

SELECT Accounts.balance

FROM Customers, Accounts

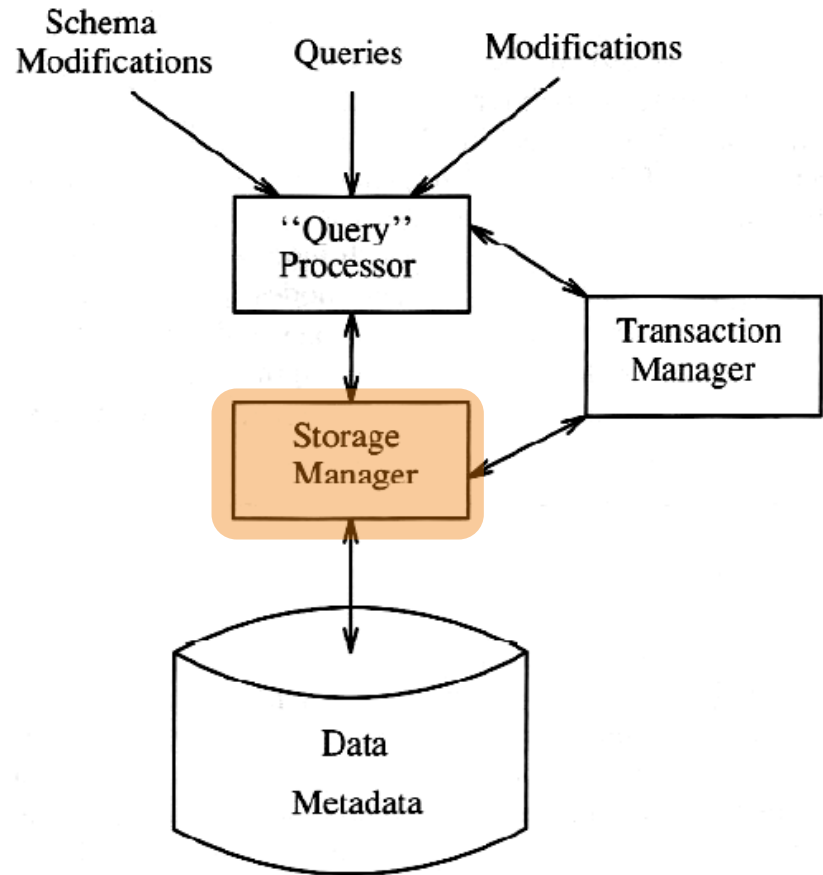
WHERE Customers.SIN = Accounts.SIN **AND** Customers.name = 'Sally';

Query processor will cleverly create a plan which **inexpensively**:

- Retrieves the tuple for “Sally” and gets the SIN number
- Retrieves the account tuples for this SIN number

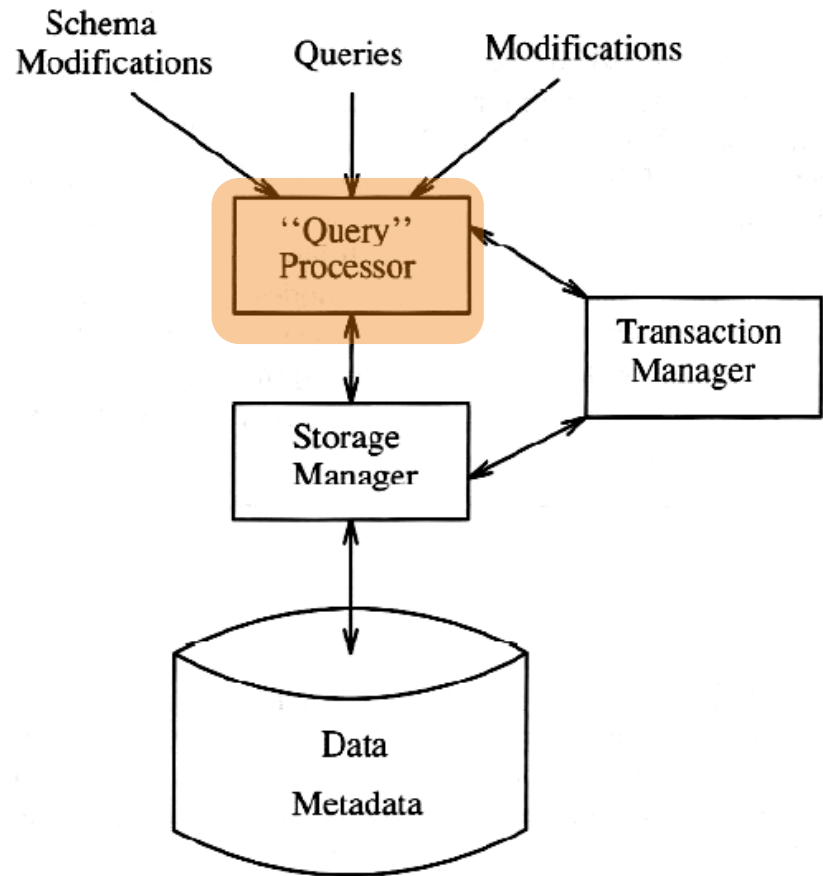
RDBMS: data consistency

- Write Ahead Logging – full recovery from failures



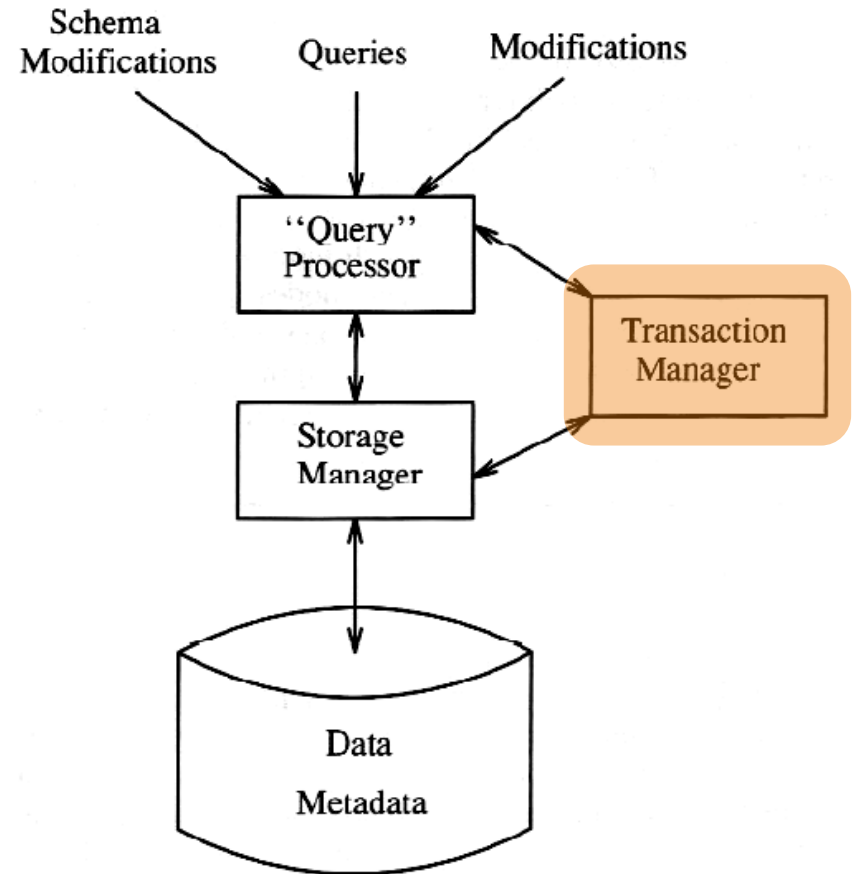
RDBMS: efficient query implementation

- Implements each operation using the most efficient EM algorithm
- Computes the best way to carry out a requested operation using relational algebra and statistics



RDBMS: concurrent execution

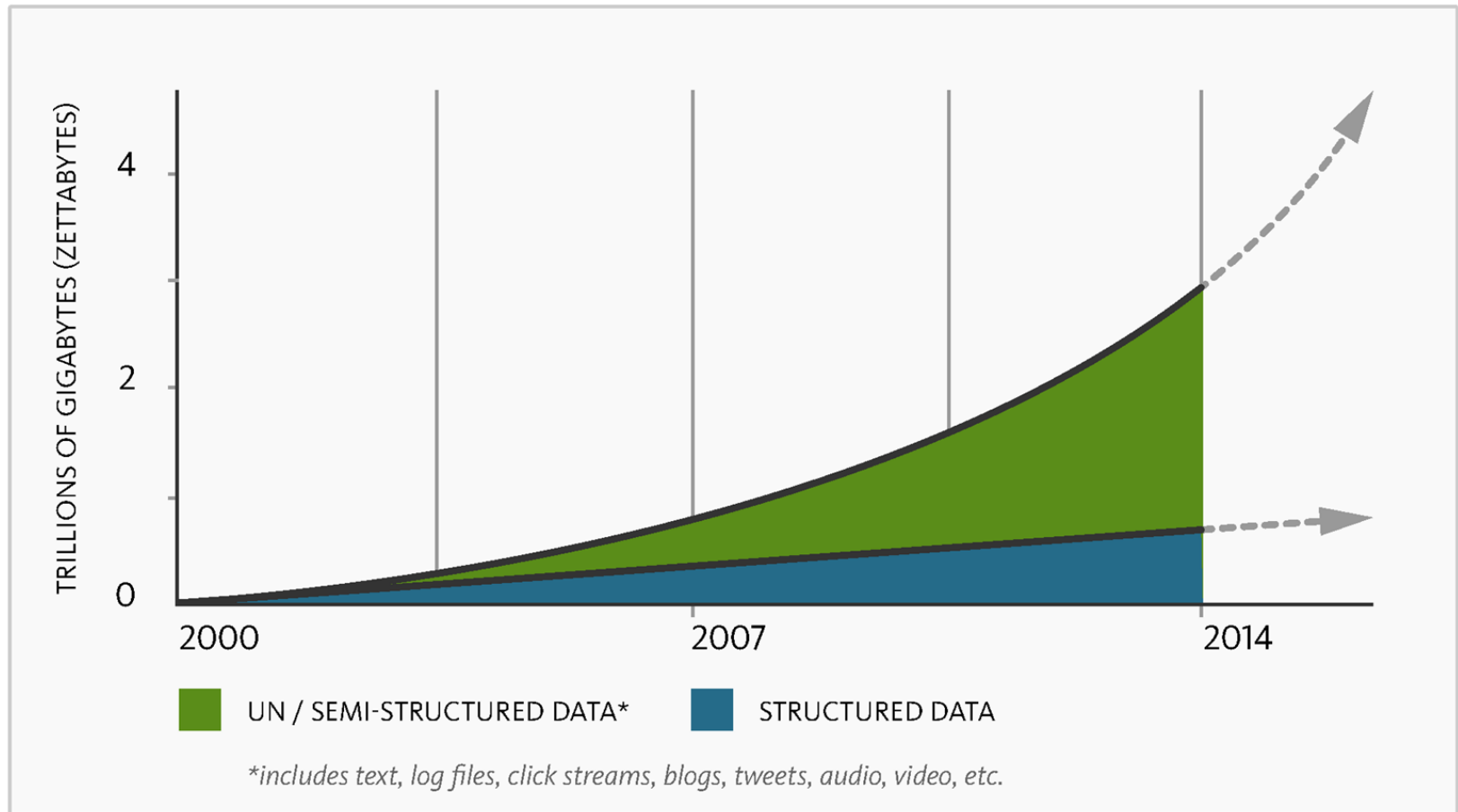
- Assures that several queries running simultaneously do not interfere with each other and that the system will not end up in an inconsistent state



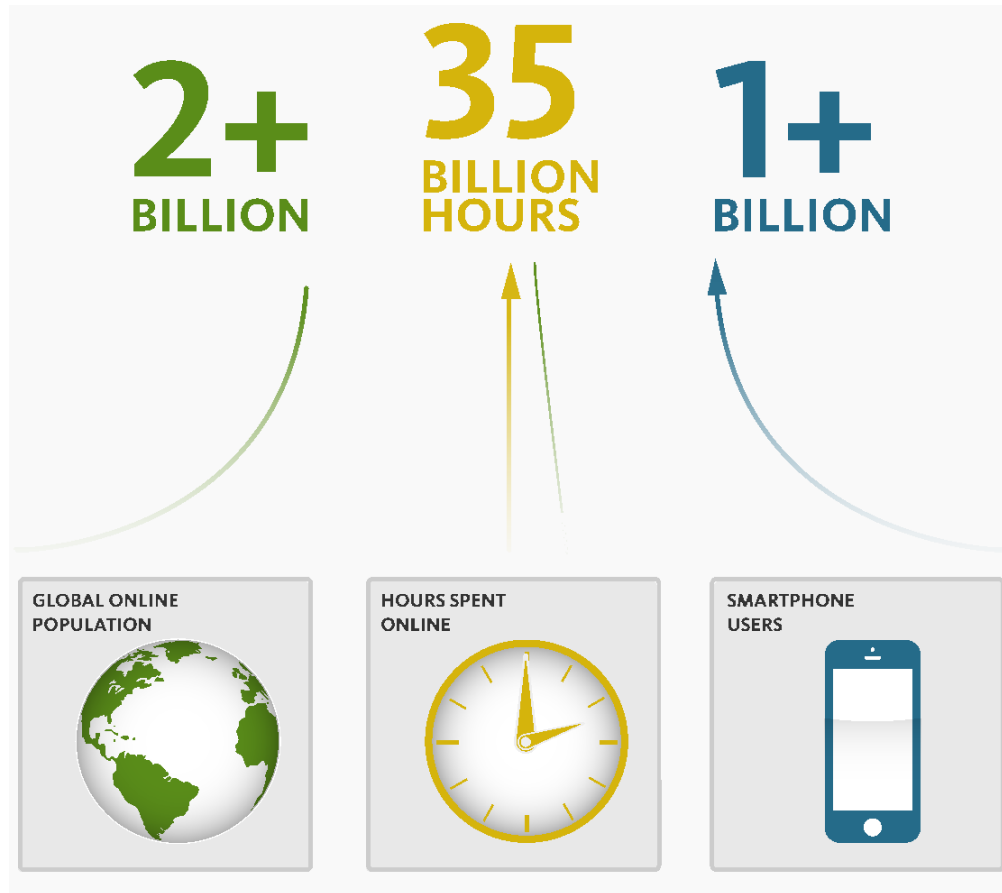
RDBMS is a very complex system

Good news: it has been already implemented for you to use

Current Trends: Big Data



Current Trends: Lots of traffic



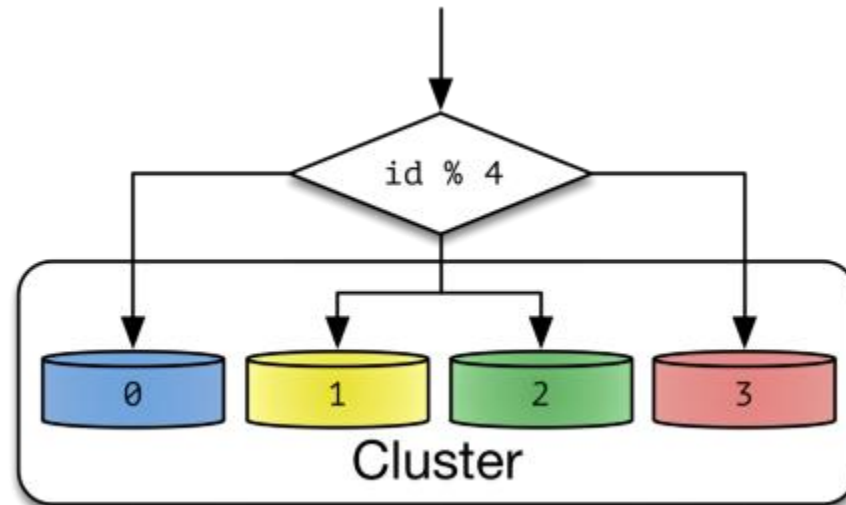
Current Trends: Cloud Computing



Scaling up

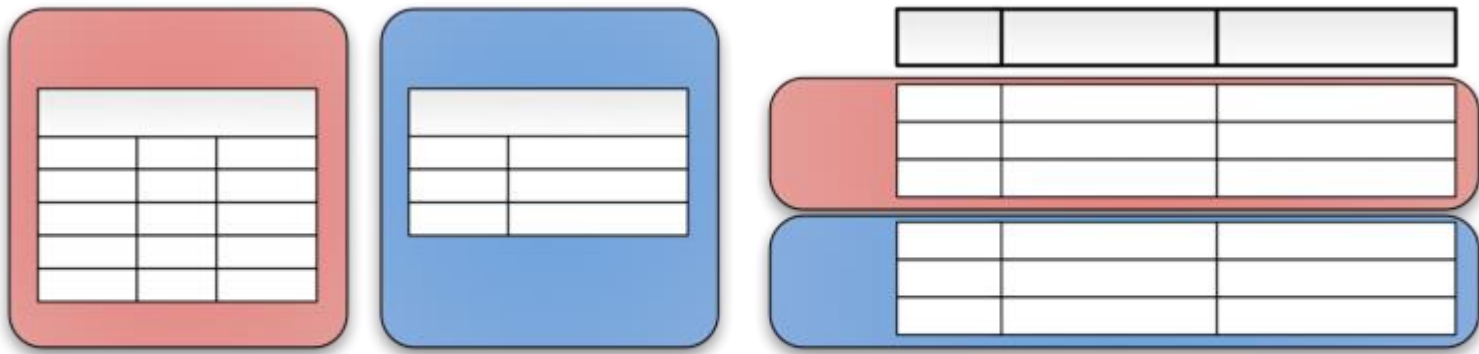
Two alternatives:

- Bigger servers
- Lots of little boxes in massive grids



Parallelism is not natural for relational databases

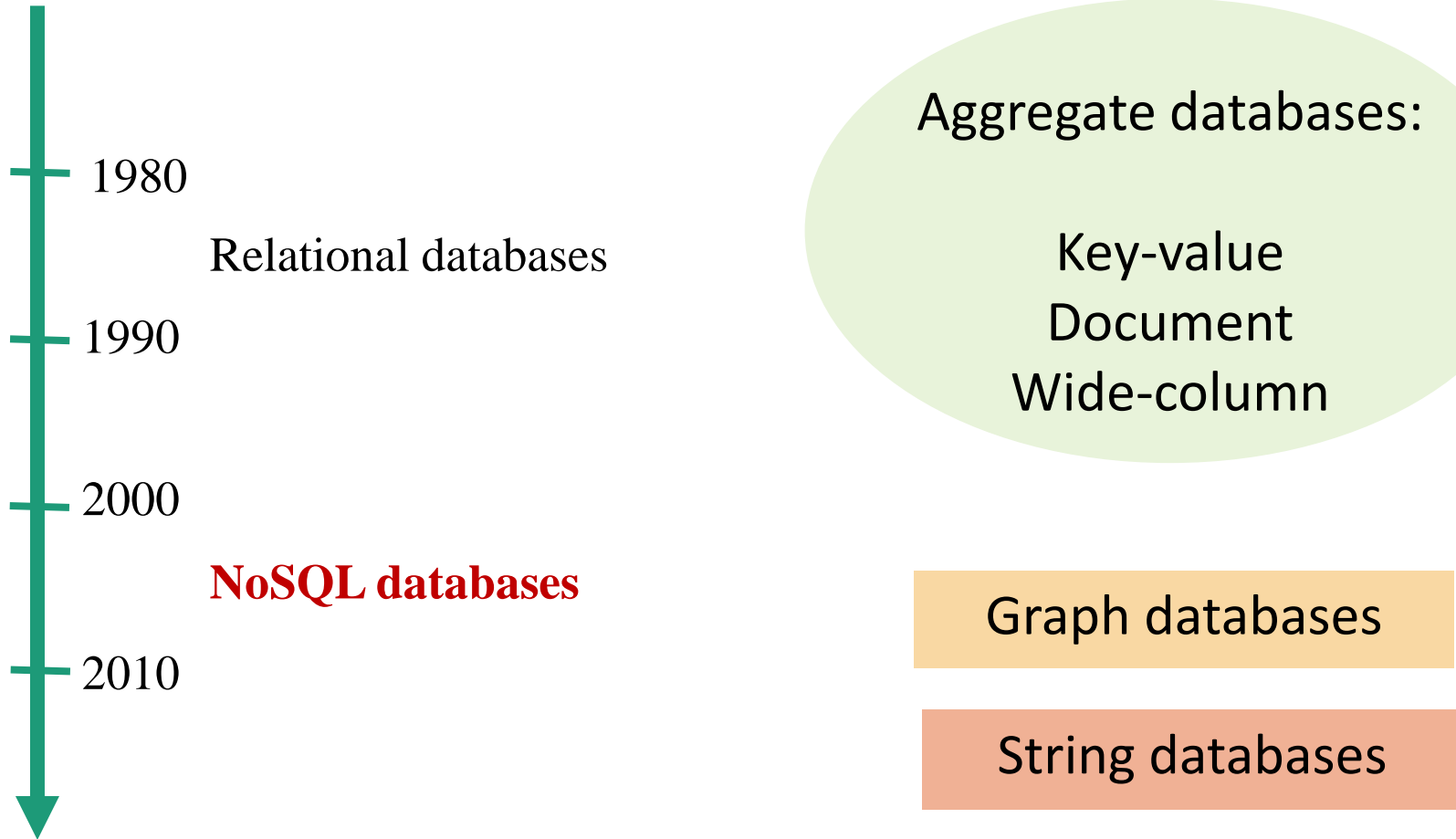
- **Vertical**: normalization, splitting into smaller tables
- **Horizontal**: splitting single table into multiple sets of rows
- SQL **designed to run as a single node**
- Both vertical partitioning and horizontal partitioning introduce performance bottlenecks



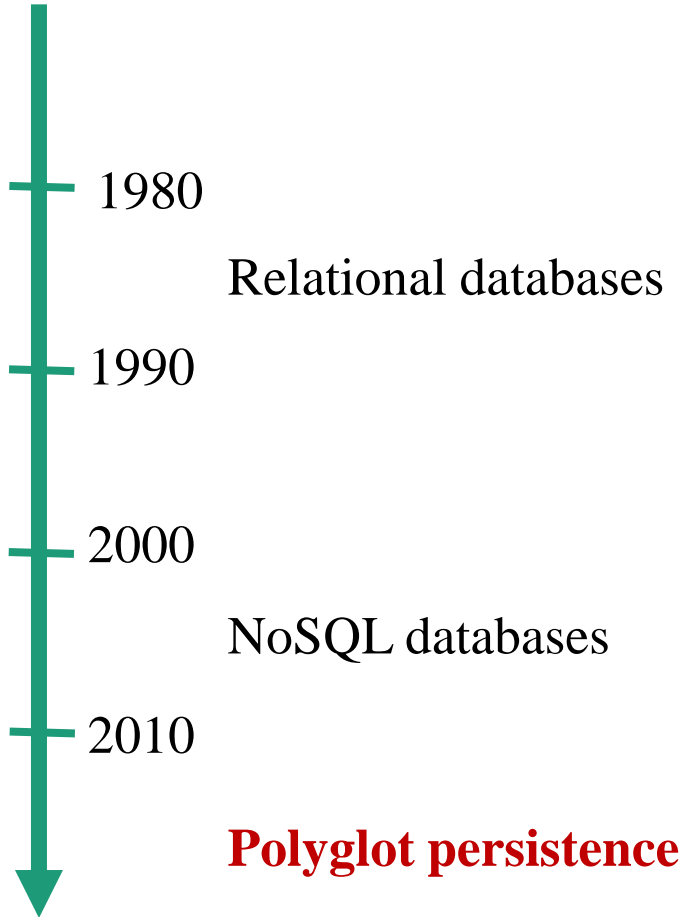
Vertical

Horizontal

History



Future?



When to use RDBMS

- Fast application development
- Data integrity and security is important
- Loss of data is unacceptable
- Concurrent data modification: by multiple users
- Data can be easily modeled as relations

When to consider alternative data stores

- String databases
- Audio, video databases
- Document databases
- Graph databases

Many facets of Database studies

- **Logical design**
 - What kinds of information to store?
 - How to *model* data?
 - How are data items connected?
- **Database programming**
 - How does one express queries on the database?
 - How is database programming combined with conventional programming?
- **Database system implementation**
 - How does one build a DBMS

In this course we explore database world from the point of view of:

Designer
Developer
User

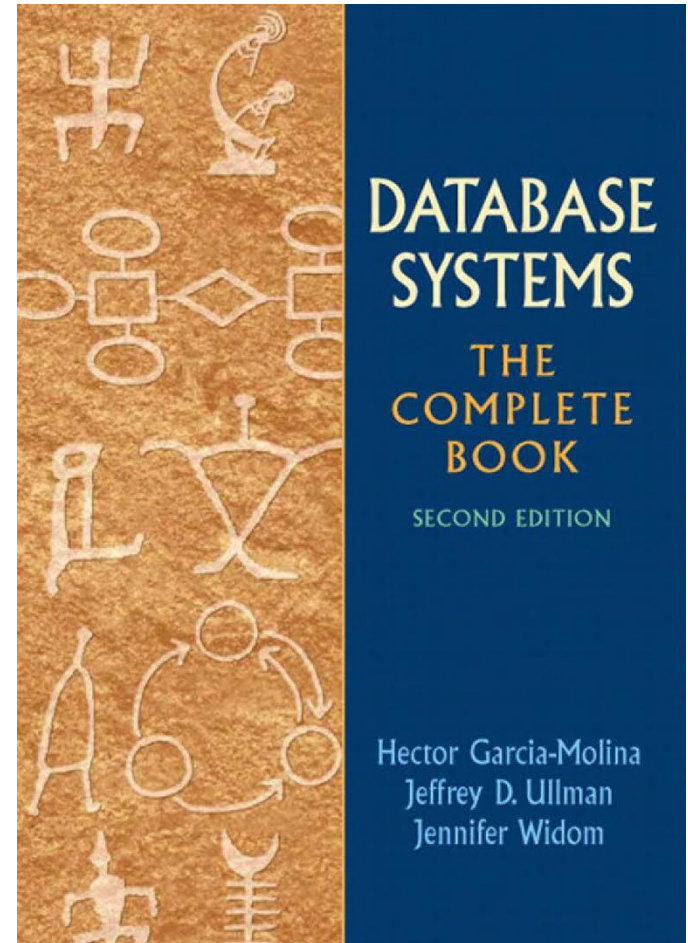
Textbook

"Database Systems: The Complete Book"

by *H. Garcia-Molina,*
J. D. Ullman,
and J. Widom,

2nd Edition.

➔ Parts I and II and some topics from Part IV



Deliverables

- Weekly homeworks: 30%
- Midterm exam: 10%
- Final project: 30% *
- Final exam: 30% *

*You need to score at least 50% on the final project and on the exam in order to pass the course

Google classroom

- Class code: mj2w0h

Sample DBMSs used in this course

- SQLite
- PostgreSQL
- Spark on IBM datascientist workbench:

<https://datascientistworkbench.com/>

You have to know how to program in Java and Python